```
USE test;

-- 1. Add a primary key to the id fields in the pets and people tables.

ALTER TABLE people
ADD PRIMARY KEY (id);

DESCRIBE pets;
DESCRIBE people;

-- 2. Add a foreign key to the owner_id field in the pets table referencing the id field
-- in the people table.

ALTER TABLE pets
ADD CONSTRAINT FK_PetsOwner
FOREIGN KEY (owner_id) REFERENCES people(id);

-- 3. Add a column named email to the people table.

ALTER TABLE people
ADD COLUMN email VARCHAR(20);

-- 4. Add a unique constraint to the email column in the people table.

ALTER TABLE people
ADD CONSTRAINT u_email UNIQUE (email);



-- 5. Rename the name column in the pets table to ‘first_name’.

ALTER TABLE pets CHANGE `name` `first_name` VARCHAR(20);

-- 6. Change the postcode data type to CHAR(7) in the addresses table.

ALTER TABLE addresses MODIFY postcode CHAR(7);

DESCRIBE addresses;
```

```
USE coffee_store;


-- 1. From the customers table, select the first name and phone number of all the females
-- who have a last name of Bluth.

SELECT first_name, phone_number FROM customers
WHERE gender = 'F'
AND last_name = 'Bluth';


-- 2. From the products table, select the name for all products that have a price greater than 3.00
-- or a coffee origin of Sri Lanka.


SELECT name, price, coffee_origin FROM products
WHERE price > 3.00
OR coffee_origin = 'Sri Lanka';


-- 3. How many male customers don’t have a phone number entered into the customers table?

SELECT * FROM customers
WHERE gender = 'M'
AND phone_number IS NULL;
```

```
USE coffee_store;

-- 1. From the products table, select the name and price of all products with a coffee origin
-- equal to Colombia or Indonesia. Ordered by name from A-Z.


SELECT name, price FROM products
WHERE coffee_origin IN ('Colombia','Indonesia')
ORDER BY name;


-- 2. From the orders table, select all the orders from February 2017 for customers with
-- id’s of 2, 4, 6 or 8.


SELECT * FROM orders
WHERE order_time BETWEEN '2017-02-01' AND '2017-02-28'
AND customer_id IN (2,4,6,8);


-- 3. From the customers table, select the first name and phone number of all customers
-- who’s last name contains the pattern ‘ar’.


SELECT first_name, phone_number, last_name FROM customers
WHERE last_name LIKE '%ar%';
```

```
USE coffee_store;

-- 1. From the customers table, select distinct last names and order alphabetically
from A-Z.

SELECT DISTINCT last_name FROM customers
ORDER BY last_name;


-- 2. From the orders table, select the first 3 orders placed by customer with id 1, in
February 2017.

SELECT * FROM orders
WHERE order_time BETWEEN '2017-02-01' AND '2017-02-28'
AND customer_id = 1
ORDER BY order_time ASC
LIMIT 3;


-- 3. From the products table, select the name, price and coffee origin but rename
the price to
-- retail_price in the results set.

SELECT name, price AS retail_price, coffee_origin FROM products;
```

```
USE coffee_store;

--  1. Select the order id and customers phone number for all orders of product id 4.

SELECT o.id, c.phone_number FROM orders o
JOIN customers c ON o.customer_id = c.id
WHERE o.product_id = 4;

--  2. Select the product name and order time for filter coffees sold between January 15th 2017 and February 14th 2017.

SELECT p.name, o.order_time FROM products p
JOIN orders o ON p.id = o.product_id
WHERE p.name = 'Filter'
AND o.order_time BETWEEN '2017-01-15' AND '2017-02-14';

-- 3. Select the product name and price and order time for all orders from females in January 2017.

SELECT p.name, p.price, o.order_time FROM products p
JOIN orders o ON p.id = o.product_id
JOIN customers c ON o.customer_id = c.id
WHERE c.gender = 'F'
AND o.order_time BETWEEN '2017-01-01' AND '2017-01-31';
```

```
USE cinema_booking_system;

-- 1. How many bookings did customer id 10 make in October 2017.

SELECT COUNT(*) FROM bookings
WHERE customer_id = 10;


-- 2. Count the number of screenings for Blade Runner 2049 in October 2017.

SELECT COUNT(*) FROM screenings s
JOIN films f ON s.film_id = f.id
WHERE f.name = 'Blade Runner 2049';


-- 3. Count the number of unique customers who made a booking for October 2017.

SELECT COUNT(DISTINCT(customer_id)) FROM bookings;
```

```
USE cinema_booking_system;

-- 1. Select the customer id and count the number of reserved seats grouped by customer for
-- October 2017.

SELECT b.customer_id, COUNT(rs.id) FROM bookings b
JOIN reserved_seat rs ON b.id = rs.booking_id
GROUP BY b.customer_id;

-- 2. Select the film name and count the number of screenings for each film that is over
-- 2 hours long.

SELECT f.name, f.length_min, COUNT(s.id) FROM films f
JOIN screenings s ON f.id = s.film_id
GROUP BY f.name, f.length_min
HAVING f.length_min > 120;
```

```
USE cinema_booking_system;

-- 1. Select the film name and length for all films with a length greater than the
average film length.

SELECT name, length_min FROM films
WHERE length_min >
(SELECT AVG(length_min) FROM films);

SELECT AVG(length_min) FROM films;

-- 2. Select the maximum number and the minimum number of screenings for a
particular film.

SELECT MAX(id), MIN(id) FROM
(SELECT film_id, COUNT(id) AS id FROM screenings
GROUP BY film_id) a;

-- 3. Select each film name and the number of screenings for that film.

SELECT name,
(SELECT COUNT(id) FROM screenings
WHERE film_id = f.id
)
FROM films f;
```

```
use cinema_booking_system;

-- Concatenate the film names and length from the films table.

SELECT CONCAT(name,": ",length_min) AS film_info FROM films;

-- Extract the customers email from the 5th character onwards.

SELECT SUBSTRING(email,5) AS email_short FROM customers;

-- Select the customers first name in lower case and their last name in upper case
-- for each customer with a last name of ‘Smith’.

SELECT LOWER(first_name) AS first_name, UPPER(last_name) AS last_name
FROM customers
WHERE last_name = 'Smith';

-- Select the last 3 letters of each film name from the films table.

SELECT SUBSTRING(name,-3) AS film_name FROM films;

-- Concatenate the first three letters in the first_name and last_name columns
together
-- from the customers table.

SELECT CONCAT(SUBSTRING(first_name,1,3)," ",SUBSTRING(last_name,1,3))
AS short_name
FROM customers;
```

```
-- Select the film id and start time from the screenings table for the date of 20th of October 2017.

SELECT film_id, start_time FROM screenings
WHERE DATE(start_time) = '2017-10-20';

-- Select all the data from the screenings table for the start time between the 6th and 13th of
-- October 2017.

SELECT * FROM screenings
WHERE DATE(start_time) BETWEEN '2017-10-06' AND '2017-10-13';

-- Select all the data from the screenings table for October 2017.

SELECT * FROM screenings
WHERE MONTH(start_time) = '10'
AND YEAR(start_time) = '2017';
```

-- Which films are over 2 hours long?

```
SELECT name, length_min FROM films
WHERE length_min > 120;
```

-- Which film had the most screenings in October 2017

```
SELECT f.name, COUNT(s.film_id) AS showings FROM screenings s
JOIN films f ON f.id = s.film_id
GROUP BY film_id
ORDER BY showings DESC
LIMIT 1;
```

```
-- How many bookings did the film ‘Jigsaw’ have in October 2017

SELECT COUNT(*) AS no_bookings FROM bookings
WHERE screening_id IN
(SELECT id FROM screenings
WHERE film_id = 5);
```

-- Which 5 customers made the most bookings in October 2017

```
SELECT c.first_name, c.last_name, COUNT(b.id) AS no_bookings FROM bookings
b
JOIN customers c ON c.id = b.customer_id
GROUP BY c.first_name, c.last_name
ORDER BY no_bookings DESC
LIMIT 5;
```

-- Which film was shown in the Chaplin room most often in October 2017

```
SELECT * FROM films;
SELECT * FROM rooms;
SELECT * FROM screenings;

SELECT f.name, COUNT(r.name) AS no_screenings FROM films f
JOIN screenings s ON f.id = s.film_id
JOIN rooms r ON r.id = s.room_id
WHERE r.id = 1
GROUP BY f.name
ORDER BY no_screenings DESC
LIMIT 1;
```

```
-- How many of the customers made a booking in October 2017

SELECT COUNT(*) FROM customers;

SELECT * FROM bookings;

SELECT COUNT(DISTINCT(customer_id)) AS no_of_customers FROM bookings;
```